API 2.0代码生成器

|  |  |
| --- | --- |
| 模块名称 | API 2.0 代码生成器 |
| 所属系统 | API2.0 |
| 模块负责人 | 陈楚(chenchu01) |
| 项目负责人 | 陈楚(chenchu01) |
| 文档提交日期 | 2016.3.15 |

百度在线网络技术（北京）有限公司

(版权所有,翻版必究)

修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | 修改后  版本号 | 修改内容简介 | 修改日期 | 修改人 |
| 1 | V1.0 | 全文 | 2016.3.15 | 陈楚 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 功能描述 1](#_Toc458115546)

[2 技术选型 1](#_Toc458115547)

[3 设计实现 1](#_Toc458115548)

[3.1 语法树解析及格式检查 2](#_Toc458115549)

[3.1.1 default语法解析 3](#_Toc458115550)

[3.1.2 enum语法解析 4](#_Toc458115551)

[3.1.3 type语法解析 4](#_Toc458115552)

[3.1.4 api语法解析 5](#_Toc458115553)

[3.2 type依赖检查 5](#_Toc458115554)

[3.3 Json doc填充 5](#_Toc458115555)

[3.4 自定义模板应用及代码输出 5](#_Toc458115556)

[4 设计评审意见 6](#_Toc458115557)

[5 附件及参考资料 6](#_Toc458115558)

# 功能描述

API 2.0代码生成器解析json doc文件，进行格式判定并根据规则生成指定的代码文件，提升研发效率。

代码生成器需要支持的功能：

* 自动检测json doc文件格式，对错误进行提示。
* 可以自定义生成规则，满足不同APP的需求及差异。
* 方便未来扩展，支持引入新类型、新定义及custom key解析。

# 技术选型

**PHP脚本+Smarty模板**。

其中PHP使用 5.2语言特性开发，并不依赖任何框架，包括odp、bingo等，不依赖core以外的PHP扩展；Smarty使用v3版本，不依赖非预装插件（以官网下载版本为准）。

这样选择的目的：

1. 最大程度与线上的PHP 5.2.17 + Smarty技术栈保持统一，减少学习成本。
2. 保障代码能够独立运行，仅依赖原始PHP环境，方便部署于本机、开发机或是测试机，并能够以命令行或者是web服务方式对外暴露。

# 设计实现

程序整体执行流程如下。

正常执行流程分为6步。

前两步里，程序会根据配置信息获取json document的存放路径，遍历其中每一个json文件并读取至内存，方便后续使用。

## 语法树解析及格式检查

在该步骤里，所有读入的json doc文件会从PHP实现形式的array转向object，并会建立起object的一棵对象树。

程序需要对每一个支持的基本类型，如int32、int64、string等实现一个对应的解析类，对传入的PHP array进行解析。对List、Type类型页会有对应的解析类，同时会有children子定义与parent父节点的双向引用。

一个可行的定义case如下：

|  |
| --- |
| class Define  function parseArr(array)  Define& parent  Hash children  function toArray() |

此时，对于输入的array，若不满足类定义则可以直接抛出异常进入异常流程。

如果该类型存在子定义，则遍历子定义数组，对每一个元素构造并解析一个新的对象。从根节点起构造对象，最终能够完整的构造完整对象树。

构造流程如下图所示。



根据API 2.0的定义，json doc需要包含：default黑话定义，type基础类型定义，enum枚举型定义，api接口定义4个部分。

语法定义存在如下依赖：（A --> B代表A依赖B）

|  |
| --- |
| type --> default  type --> enum  type --> type  api --> type  api --> default  api --> enum |

为了保障基本的依赖关系，程序解析流程按顺序分为如下几个子流程。

### default语法解析

遍历所有的default定义，分别解析并存储全局的default list，建立key到object的对应关系。解析过程中出错或是key重复均会导致异常并退出。

完成default语法解析后，程序内存中将包含hash<key, object> default\_list，并供后续使用。

### enum语法解析

与default类似，enum语法解析会建立全局的enum list，其中为enum key对应到object。解析过程出错或是enum key重复均会导致异常并退出。

与default不同的是，enum支持后续在type与api解析过程中动态添加新的enum key。所以内存中的enum\_list会持续扩容直到整个解析结束。

### type语法解析

type解析最大的难点在于base语法，即继承。由于type存在自依赖，即typeA依赖typeB，甚至是多级依赖（多级继承），故在**解析过程中不判断type的自依赖关系**。Type中的子定义可能会依赖于default或是enum，此时会查询对应list里的object，若不存在，则解析失败。

Type解析会存储type name与object的对应关系，并产生type\_list。与enum\_list类似，type\_list也会动态的扩充直到整个解析结束。

整体流程如下图：



### api语法解析

由于API分为input、output、type3个域，在解析过程中会对这3个域分别解析。

程序需要优先解析并存储type域，保障后续解析input、output时能够顺利引用。Type的解析与上一节中的type解析流程相同。

在3个区域都完整解析后，程序需要在内存中建立一个api object，同时建立hash<api name, api object> api\_list方便后续使用。同样的，api name不允许重复。

## type依赖检查

由于type存在自依赖，故在所有type均解析完毕后需要进行依赖检查。

程序实现如下：

1. 将所有type置入uncheck list，并新建一个hash checked\_list存放已经通过校验，即依赖检查通过的type。
2. 初始化T=0。
3. 循环遍历每一个uncheck list item。若对应的type依赖为空（base为空）或依赖的type已在checked\_list内，则将该type移动到checked\_list中，并计数T=T+1。
4. 遍历结束，若此时uncheck list为空，则所有type均通过了依赖检查。此时依赖检查返回正常。
5. 若uncheck list不为空，则检查T。当T > 0时，回到步骤2重新执行。若T == 0则依赖检查失败并报错。剩余的type均为异常类型。

## Json doc填充

经过前几个步骤后，所有的default、enum、type及api均已被维护为object并分别存放在对应的list中。为了后续Smarty模板使用方便，需要将所有的object重新填充回array并完全展开。

此时仅需要对所有的根元素调用toArray方法即可完成。

## 自定义模板应用及代码输出

为了方便多端扩展及二次开发，程序使用Smarty模板进行输出。通过对应的PHP controller来调用Smarty的渲染方法，使用上一步中获得的Json doc充当模板变量即可。

# 设计评审意见

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | 问题描述 | 提出人 | 处理方式/说明 | 状态 |
| 1 |  |  |  | Open |
| 2 |  |  |  | Close |
| 3 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

这里累计记录和跟踪该文档评审时发现的问题。

# 附件及参考资料

填写文档相关的附件或参考资料。若是不常变更的文档（比如调研报告），建议以对象方式插入到该文档中。如是经常变更的文档（比如接口文档），建议在此列出文件名即可。